The Caesar Cipher

Now is the time! Plaintext
Opx jt uif ujnf! Ciphertext

This is the Caesar Cipher with a key of 1 (B).

Plaintext

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|  0 |  1 |  2 |  3 |  4 |  5 |  6 |  7 |  8 |  9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

Ciphertext

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|  0 |  1 |  2 |  3 |  4 |  5 |  6 |  7 |  8 |  9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

Enciphering is performed by replacing each plaintext character with the character “one down” in the alphabet.

The Caesar Cipher is a substitution cipher.

Attacking the Cipher

- Exhaustive search
  - If the key space is small enough, try all possible keys until you find the right one
  - Caesar cipher has 26 possible keys
- Statistical analysis
  Compare to 1-gram model of English. If 1-gram frequencies match those of English but the other n-gram frequencies do not, it is probably a substitution cipher.

Caesar’s Problem

- Key is too short:
  - Can be found by exhaustive search
  - Statistical frequencies not concealed well
  - They look too much like regular English letters
- So make it longer
  - Multiple letters in key
  - Idea is to smooth the statistical frequencies to make cryptanalysis harder

Vigenère Cipher

- Like Caesar cipher, but use a phrase for the key
- Example:
  - Message THE BOY HAS THE BAG
  - Key BCD (1, 2, and 3 letters down)
  - Encipher using Caesar cipher for each letter:
    key BCD BCD BCD BCD
    plain THE BOY HAS THE BAG
    cipher U J H C Q B J C S U J H C C J
- The Vigenère cipher is a polyalphabetic substitution cipher.

Vigenère Tableau

<table>
<thead>
<tr>
<th>BCD BCD BCD BCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B C D N O P Q</td>
</tr>
<tr>
<td>B C D E O P Q R</td>
</tr>
<tr>
<td>C D E F P Q R S</td>
</tr>
<tr>
<td>D E F G Q R S T</td>
</tr>
<tr>
<td>E F G H R S T U</td>
</tr>
<tr>
<td>F G H I S T U V</td>
</tr>
<tr>
<td>G H I J T U V W</td>
</tr>
<tr>
<td>H I J K U V W X</td>
</tr>
<tr>
<td>I J K L V W X Y</td>
</tr>
<tr>
<td>J K L M W X Y Z</td>
</tr>
<tr>
<td>K L M N X Y Z A</td>
</tr>
<tr>
<td>L M N O Y Z A B</td>
</tr>
<tr>
<td>M N O P Z A B C</td>
</tr>
</tbody>
</table>
**A “Real” Vigenère Tableau**

<table>
<thead>
<tr>
<th>TWASBRILLIGANTHESLITHTYTOVES</th>
<th>TWASBRILLIGANTHESLITHTYTOVES</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>TWASBRILLIGANTHESLITHTYTOVES</td>
<td>TWASBRILLIGANTHESLITHTYTOVES</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>TWASBRILLIGANTHESLITHTYTOVES</td>
<td>TWASBRILLIGANTHESLITHTYTOVES</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>UXBSTCUMJHIBESTUFTMUIZUPWFT</td>
<td>UXBSTCUMJHIBESTUFTMUIZUPWFT</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>YYCUDTJKNNIKTCICFYYJUGUNKJAVQXG</td>
<td>YYCUDTJKNNIKTCICFYYJUGUNKJAVQXG</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>WXIVSTLDOIJDGQWHIVGOLWKSBRHYV</td>
<td>WXIVSTLDOIJDGQWHIVGOLWKSBRHYV</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>XAEFVYMPPKEMHEKXLTWMPMLCXSZTWIN</td>
<td>XAEFVYMPPKEMHEKXLTWMPMLCXSZTWIN</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>YBFXGMNQNLFSYJMJXQNYMDYTAJX</td>
<td>YBFXGMNQNLFSYJMJXQNYMDYTAJX</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>ZCGYHORROMGTMJZSNYRDZSNZUKXY</td>
<td>ZCGYHORROMGTMJZSNYRDZSNZUKXY</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>HADZITYPSSPNHKAKOZUPZPAFVAUCLZ</td>
<td>HADZITYPSSPNHKAKOZUPZPAFVAUCLZ</td>
</tr>
<tr>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>IJKLMTVNEVVVFBOOLMNHQJKWNBQCKMK</td>
<td>IJKLMTVNEVVVFBOOLMNHQJKWNBQCKMK</td>
</tr>
<tr>
<td>J</td>
<td>J</td>
</tr>
<tr>
<td>JKFJEKARUPFJWMCQDSBQRQHCKNEB</td>
<td>JKFJEKARUPFJWMCQDSBQRQHCKNEB</td>
</tr>
<tr>
<td>K</td>
<td>K</td>
</tr>
<tr>
<td>KJEFJEBJEBJEBJEBJEBJEBJEBJEB</td>
<td>KJEFJEBJEBJEBJEBJEBJEBJEBJEB</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>LEBLCKYTVWFTMYOEFDFTZFEZGZPFD</td>
<td>LEBLCKYTVWFTMYOEFDFTZFEZGZPFD</td>
</tr>
<tr>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>MLEHNSXKGZKMMKZKMMKZKMMKZKMM</td>
<td>MLEHNSXKGZKMMKZKMMKZKMMKZKMM</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>NREUQHTXYESELAWLWMPAOHRS</td>
<td>NREUQHTXYESELAWLWMPAOHRS</td>
</tr>
</tbody>
</table>

**A Generic Vigenère Tableau**

<table>
<thead>
<tr>
<th>TWASBRILLIGANTHESLITHTYTOVES</th>
<th>TWASBRILLIGANTHESLITHTYTOVES</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>TWASBRILLIGANTHESLITHTYTOVES</td>
<td>TWASBRILLIGANTHESLITHTYTOVES</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>UXBSTCUMJHIBESTUFTMUIZUPWFT</td>
<td>UXBSTCUMJHIBESTUFTMUIZUPWFT</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>YYCUDTJKNNIKTCICFYYJUGUNKJAVQXG</td>
<td>YYCUDTJKNNIKTCICFYYJUGUNKJAVQXG</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>WXIVSTLDOIJDGQWHIVGOLWKSBRHYV</td>
<td>WXIVSTLDOIJDGQWHIVGOLWKSBRHYV</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>XAEFVYMPPKEMHEKXLTWMPMLCXSZTWIN</td>
<td>XAEFVYMPPKEMHEKXLTWMPMLCXSZTWIN</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>YBFXGMNQNLFSYJMJXQNYMDYTAJX</td>
<td>YBFXGMNQNLFSYJMJXQNYMDYTAJX</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>ZCGYHORROMGTMJZSNYRDZSNZUKXY</td>
<td>ZCGYHORROMGTMJZSNYRDZSNZUKXY</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>HADZITYPSSPNHKAKOZUPZPAFVAUCLZ</td>
<td>HADZITYPSSPNHKAKOZUPZPAFVAUCLZ</td>
</tr>
<tr>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>IJKLMTVNEVVVFBOOLMNHQJKWNBQCKMK</td>
<td>IJKLMTVNEVVVFBOOLMNHQJKWNBQCKMK</td>
</tr>
<tr>
<td>J</td>
<td>J</td>
</tr>
<tr>
<td>JKFJEKARUPFJWMCQDSBQRQHCKNEB</td>
<td>JKFJEKARUPFJWMCQDSBQRQHCKNEB</td>
</tr>
<tr>
<td>K</td>
<td>K</td>
</tr>
<tr>
<td>KJEFJEBJEBJEBJEBJEBJEBJEBJEB</td>
<td>KJEFJEBJEBJEBJEBJEBJEBJEBJEB</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>LEBLCKYTVWFTMYOEFDFTZFEZGZPFD</td>
<td>LEBLCKYTVWFTMYOEFDFTZFEZGZPFD</td>
</tr>
<tr>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>MLEHNSXKGZKMMKZKMMKZKMMKZKMM</td>
<td>MLEHNSXKGZKMMKZKMMKZKMMKZKMM</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>NREUQHTXYESELAWLWMPAOHRS</td>
<td>NREUQHTXYESELAWLWMPAOHRS</td>
</tr>
</tbody>
</table>

**Attacking the Cipher**

**Approach:**
- Establish period; call it $n$
- Break message into $n$ parts, each part being enciphered using the same key letter
- Solve each part
- You can leverage one part from another

This is based on the assumption that the key repeats. What if it didn’t? (It’s a one-time pad.)

**Transposition Cipher**

- Rearrange letters in plaintext to produce ciphertext
- **Example** (Rail-Fence Cipher)
  - Plaintext is **HELLOWORLD**
  - Rearrange as **HLOOL ELWRD**
  - Ciphertext is **HLOOL ELWRD**

**Attacking the Cipher**

- Rearrange letters to form $n$-grams with highest frequencies
- Ciphertext: **HLOOLELWRD**
- Frequencies of 2-grams beginning with H
  - HE 0.0305
  - HO 0.0043
  - HL, HW, HR, HD ≤ 0.0010
- Frequencies of 2-grams ending in H
  - WH 0.0026
  - EH, LH, OH, RH, DH ≤ 0.0002
- Implies E follows H

**Example**

- Arrange so the H and E are adjacent
  - HE
  - LL
  - OW
  - OR
  - LD
- Read off across, then down, to get original plaintext
Product Ciphers

- A “product cipher” is a combination of transposition and substitution.
- Applying ciphers serially does not necessarily make the encryption stronger.

Confusion and Diffusion

- Confusion: The relationship between plaintext and key should be complex.
  - The Caesar cipher has low confusion
  - A one-time pad has high confusion
- Diffusion: Information from the plaintext affects many parts of the ciphertext.
  - Changing a single plaintext character should not change only a single ciphertext character.

Overview of the DES

- A block cipher:
  - encrypts blocks of 64 bits using a 64 bit key
  - outputs 64 bits of ciphertext
  - effective key size is 56 bits
- A product cipher
  - basic unit is the bit
  - performs both substitution and transposition on the bits
- Cipher consists of 16 rounds (iterations) each with a round key generated from the user-supplied key
  - 56 bits is too short!

Double DES

- Encrypt twice, with two different keys.
- Effective key length is 57 bits (not 112.)

Triple DES

- Three keys: encrypt, decrypt, encrypt; effective key length, 112 bits.
- Two-key 3DES has an effective key length of 80 bits.

Potential Problems

- It was argued that 56 bits was too short
- Design decisions, particularly for the S-boxes, were classified. Trapdoors?
- Worrisome properties: weak and semi-weak keys, complementation property.
- Broken by differential cryptanalysis (1990); some properties of DES suggested that the designers already knew about differential cryptanalysis.
The Advanced Encryption Standard

- DES is too weak for use now
- NIST selected Rijndael as Advanced Encryption Standard, successor to DES effective in May, 2002
- Designed to withstand attacks that were successful on DES

Symmetric (Secret Key) Encryption

- DES  56 bits (1973)
- 3DES  3 keys, multiple encryption
- AES (Rijndael)  128 (and more) bit keys (2001)
- Others

The Problem with Secret Keys

- You use secret-key cryptography to protect unsecure transmission or storage
- Problem: how does the recipient get the secret key!?!

(Out of band transmission is one secure way, but often one could simply transmit the message, and not just the key, if a secure channel exists.)

Public Key Cryptography

- Also known as “asymmetric key cryptography.”
- Encrypt with one key (publicly available)
- Decrypt with a different key, known only to the recipient.
- How? Trapdoor functions!
  Functions that are easy to compute, for which the inverse is intractable.
- $293 \times 307$ is easy. Find the prime factors of $89,851$!

Public Key Cryptography

- With public key cryptography, one creates a pair of keys, $k_v$ (private) and $k_b$ (public).
- A ciphertext message, $C$ is formed from a plaintext message $M$: $C = f(k_b, M)$
- Deriving $M$ from $C$ and $k_b$ is intractable.
- Deriving $k_v$ from $k_b$ is intractable.
- However, $M = f(k_v, C)$ is straightforward to compute, but computationally intensive.
The Math of Public Key Cryptography

Choose two large prime numbers, \( p \) and \( q \). Let \( n = pq \); let \( \varphi = (p-1)(q-1) \).

Choose \( e \) such that \( 1 < e < n \) and \( e \) and \( \varphi \) are relatively prime (have no common factors).

Compute \( d \) such that \( ed \mod \varphi = 1 \).

The public key is: \( k_b = (e, n) \)

The private key is: \( k_v = (d, n) \)

Example

Choose \( p = 5, q = 11 \) (both prime) \n\( n = pq = 55 \) \n\( \varphi = (p-1)(q-1) = 40; \) let \( e = 9 \) (relatively prime to \( \varphi \)) \nd will be 49 \( (9 \times 49 = 441; \ 441 \mod 40 = 1) \)

Let \( m \), the message, be 6 (Must be less than \( n \))

Encrypt: \( c = m^e \mod n = 6^9 \mod 55 = 46 \)

Decrypt: \( m = c^d \mod n = 46^{49} \mod 55 \)

(46^{49} is an 82-digit number!)

To break: Factor \( n \) to obtain \( p, q \), then \( e, d \)

Another Example

- Take \( p = 7, q = 11 \), so \( n = 77 \) and \( \varphi(n) = 60 \)
- Alice chooses \( e = 17 \), making \( d = 53 \)
- Bob wants to send Alice secret message HELLO (07 04 11 11 14)
- 07^{17} mod 77 = 28
- 04^{17} mod 77 = 16
- 11^{17} mod 77 = 44
- 14^{17} mod 77 = 44
- Bob sends 28 16 44 44 42

Decrypting the Example

- Alice receives 28 16 44 44 42
- Alice uses private key, \( d = 53 \), to decrypt message:
  - 28^{53} mod 77 = 07
  - 16^{53} mod 77 = 04
  - 44^{53} mod 77 = 11
  - 44^{53} mod 77 = 11
  - 42^{53} mod 77 = 14
- Alice translates message to letters to read HELLO
- No one else could read it, as only Alice knows her private key and that is needed for decryption

Computational Effort

- Public key encryption can be 10,000 times more computational work than secret key encryption.
- Why? We are dealing with difficult arithmetic (exponentiation) on very large (hundreds of digits) numbers.
**Public Key Digital Signature**
Public key cryptography works both ways; the keys are inverses of one another!
Bob encrypts a message with his **private** key. *Anyone* can decrypt it using Bob’s **public** key. Only Bob could have encrypted it, so, it is “signed” by Bob.
Uhhh, but *anyone* can decrypt it; there is no confidentiality. (This is exactly analogous to a handwritten signature.)

**Cryptographic Checksums**
- Idea: Detect tampering with messages.
- Needed: A function that “characterizes” a message in some way.
- A strong hash function:
  - $h(x)$ is easy to compute
  - Deriving $x$ from $h(x)$ is infeasible (protects confidentiality)
  - Given $x$, finding $x'$ such that $h(x') = h(x)$ is infeasible (protects integrity by resisting collision attacks)

**Hash Functions**
- **DOG**
  - hex 44 4f 47
  - Hash = 44+4f+47 = da
- **BOG**
  - hex 42 4f 47
  - Hash = 42+4f+47 = d8
Different hash codes imply different messages.
Does the same hash code mean that the messages are identical?

**The MD5 and SHA Hashes**
- The MD5 hash code is 128 bits; SHA is 160.
- So, there are $2^{128}$ or $2^{160}$ possible hashes (a lot!)
- The probability that two messages have the same hash code is very small.
- Can a message be especially crafted to have the same hash code as an arbitrary message? (Not easily.)
- That is the strength of the MD5 and SHA algorithms.

**Collision Attack**
A **collision attack** is an attack that attempts to create a message with the same hash code as some other message.
A source program has a given MD5 hash.
Can you insert malicious code in the program, then change it in some other way such that it has the same MD5 hash as the “good” program?

**Digital Signature Improved**
Bob computes a “digest” of a message using a hash function; this is smaller than the message and of fixed size, but has the property that changing the message changes the computed digest.
Bob encrypts the digest with his private key.
Anyone can decrypt the digest using Bob’s public key, but only Bob could have encrypted it. The message is signed.
If the digest matches the message, the message has not been altered.
**Digital Signature**

Bob computes a digest over a message and encrypts it with his **private** key. Bob encrypts message and digest with Betty's **public** key. Only Betty can decrypt the message because it needs her private key. The message is confidential. She decrypts and verifies the digest using Bob's public key. The message is signed.

**Digital Signature**

If the digests are equal, the sender is assured and no tampering has occurred.

---

**Digital Signature with Confidentiality**

Bob computes a digest over a message and encrypts it with his **private** key. Bob encrypts message and digest with Betty's **public** key. Only Betty can decrypt the message because it needs her private key. The message is confidential. She decrypts and verifies the digest using Bob's public key. The message is signed.

---

**Digital Signature**

Recipient's **private** key

Encrypted, Digitally-Signed Message

**Digital Signature**

Recipient's **public** key

Encrypted, Digitally-Signed Message

Decrypted Digest = Computed Digest

If the digests are equal, the sender is assured and no tampering has occurred.
Security Services

- Confidentiality
  - Only the owner of the private key knows it, so text enciphered with public key cannot be read by anyone except the owner of the private key.
- Integrity
  - Enciphered messages cannot be changed undetectably without knowing private key. (Data integrity.)
  - Message enciphered with private key came from someone who knew it. (Source integrity and non-repudiation.)

Public Keys with Less Computation

- Public key encryption is time-consuming... up to 10,000 times more work than secret key encryption.
- Solution: generate a random, one-time secret key.
- Encrypt the message with the secret key.
- Encrypt the secret key with the recipient’s public key.
- Send encrypted message and encrypted key.

Hybrid Cryptography

Message

Random

Session Key

Encrypted Message

The session key is a shared key; computation for encryption is (relatively) easy.

Packaging a Hybrid Crypto Message

Message Digest

<table>
<thead>
<tr>
<th>Encrypt with</th>
<th>Session Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender’s Private Key</td>
<td>Encrypt with Recipient’s Public Key</td>
</tr>
</tbody>
</table>

Encrypted Message

| Encrypted Digest | Encrypted Key |

How SSL/TLS Works

Generate a (secret) session key at the server
Encrypt the session key with the server’s private key (to sign it) and the browser’s public key (to secure it)
Browser decrypts the session key with the server’s public key and its own private key
The session key is used for one communication session only.
**Diffie-Hellman**

- Problem: how to share a secret over an unsecure channel without public key encryption.
- Sender S and Recipient R agree on two values, \( n \) and \( g \). \( n \) is the size of \( g \). This is unencrypted, so an attacker can learn \( n \) and \( g \).
- S and R generate secret numbers \( s \) and \( r \).

- S sends \( g^s \) to R
- R sends \( g^r \) to S
- S computes \((g^r)^s = (g^s)^r\)
- R computes \((g^r)^s = (g^s)^r\)
- They are equal, so R and S have a shared secret
- An attacker must factor \( g^s \) and \( g^r \) to derive \( r, s \).

**CIA Again**

- Confidentiality
  - Encryption protects confidentiality
- Integrity
  - Hashes or encryption provide message integrity
  - Digital signatures provide for non-repudiation
  - Encryption can be used for authentication.
- Availability is not addressed by encryption

**Public Key Cryptography**

Bill wants to communicate securely with George. He gets George’s public key from a key repository and uses it to encrypt his message.

- Bob
- George

George’s Public Key

Key Repository

Bob uses George’s public key to send a message.

George decrypts the message with his private key.

**Man-in-the-Middle Attack**

Eve (the man in the middle) sneakily replaces George’s public key in the repository with her own.

- Bob
- Eve
- George

Eve can intercept the message, read it, re-encrypt it, and send it onward.

George’s Public Key

Key Repository

*But really Eve’s public key!*

**Public Key Infrastructure**

- Goal: bind identity to key
- Classical: not possible as all keys are shared
  - Use protocols to agree on a shared key (see earlier)
- Public key: bind identity to public key
  - Crucial as people will use key to communicate with principal whose identity is bound to key
  - Erroneous binding means no secrecy between principals
  - Assume principal identified by an acceptable name
Certificates
• Create token (message) containing
  • Identity of principal (here, George)
  • Corresponding public key
  • Timestamp (when issued)
  • Other information (perhaps identity of signer)
Digitally signed by trusted authority (here, Cathy)
\[ C_d = \{ e_A \parallel George \parallel T \} d_C \]

Public Key in a Certificate
If George’s public key is contained in a digital certificate, signed by someone Bob trusts, it is much harder to tamper with the key.

Bob
Key Repository
George

Brown’s Public Key

Transitive Trust
• Do you trust j.cannady@computer.org?
• If so, you can trust that you have Brown’s real key.
• No? Read on…

Do You Trust Any of These People?

Transitive Trust
• Do you trust j.cannady@computer.org?
• If so, you can trust that you have Brown’s real key.
• Do you trust any of the people who have signed Dr. Cannady’s key? If so, you can trust that you have Brown’s real key.
• This is transitive trust.
Certificate Authorities

- A CA is a trusted organization (Verisign?!) that exists to issue digital certificates.
- The CA verifies an identity…
- Creates a digital certificate with that identity and a public key…
- And signs the certificate with its own private key.
- The CA’s public key is “well-known” and can be used to verify the signature

Digital Certificates

- Version number
- Owner (Subject)
- Public key
- Issuer (CA)
- Serial number
- Validity dates
- Certificate usage
- Extensions

Lifetimes of Certificates and Keys

- Trade off: security vs. overhead
- Renewal of certificate
  - Original keys, new certificate
  - New keys and certificate

Expired Certificate

Browsers warn of expired certificates.

Questions

Key Points

- Two main types of cryptosystems: classical (shared key) and public key
- Classical cryptosystems encipher and decipher using the same key
- Public key cryptosystems encipher and decipher using different keys, but are much more computationally intensive
- Cryptographic checksums provide a check on integrity
- Cryptographic checksums and public key crypto provide for digital signatures.
- Hybrid cryptography addresses the key exchange problem and the problem of computational effort.